

Physical Computing mit Arduino

- Internetverbindung herstellen
- Inhalt von USB-Stick auf Computer kopieren (Ordner „ICT-V_Arduino“)
- Ordner Nr. 1:
Arduino IDE und
Tone Library installieren

Physical Computing mit Arduino

Ordner Nr. 2:
„Speaker_Sketch“

Sketch

Verifizieren



```
Speaker_Sketch | Arduino 1.0.6

/*
Melodie mit Lautsprecher

Tone Library muss installiert sein!
https://github.com/bogman/Tone

Speaker GND und 12
*/

#include <Tone.h> //Bibliothek(Library) Tone hinzufügen

Tone speaker; //Den Lautsprecher als Speaker festlegen

void setup ()
{
  speaker.begin(12); //Den Lautsprecher auf Pin 12 definieren
}

void loop()
{
  speaker.play( NOTE_G5, 100 ); //In Klammer: (Tonart, Tonlänge)
  delay (250); //Pause (Milliseskunden) zwischen den Tönen
  speaker.play( NOTE_C6, 100 );
  delay (250);
  speaker.play( NOTE_E6, 100 );
  delay (300);
}

Kompilierung abgeschlossen.

Binäre Sketchgröße: 2.766 Bytes (von einem Maximum von 32.256 Bytes)

1 Arduino Uno on /dev/tty.usbmodem1411
```

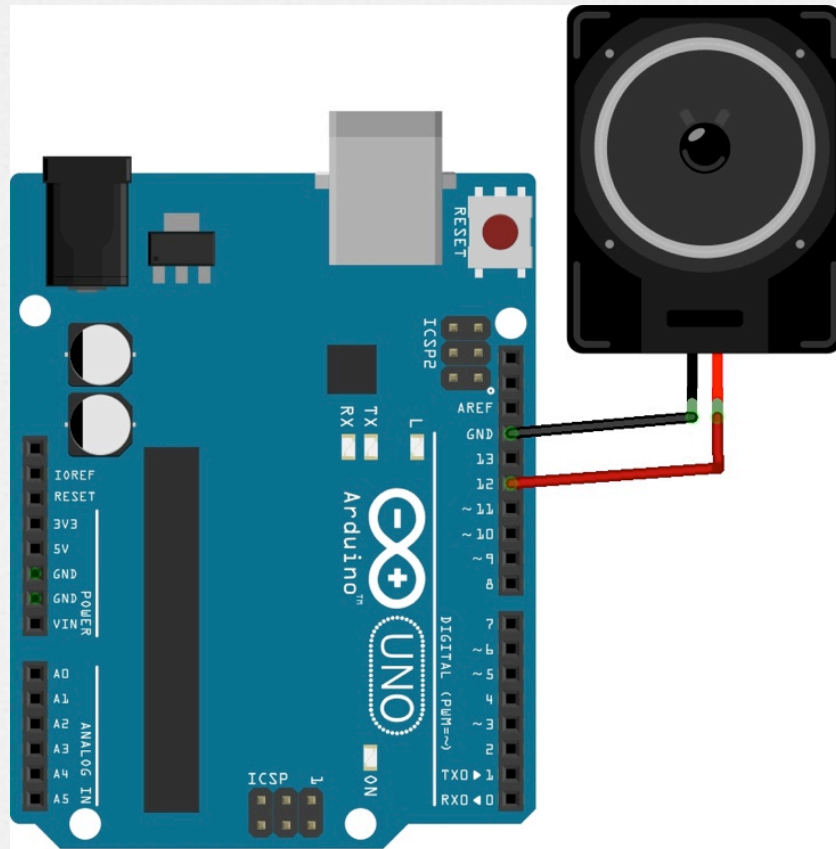

Physical Computing mit Arduino

Lautsprecher an
Arduino
anschiessen:

- auf GND
- + auf 12

Arduino und
Computer mit USB
Kabel verbinden

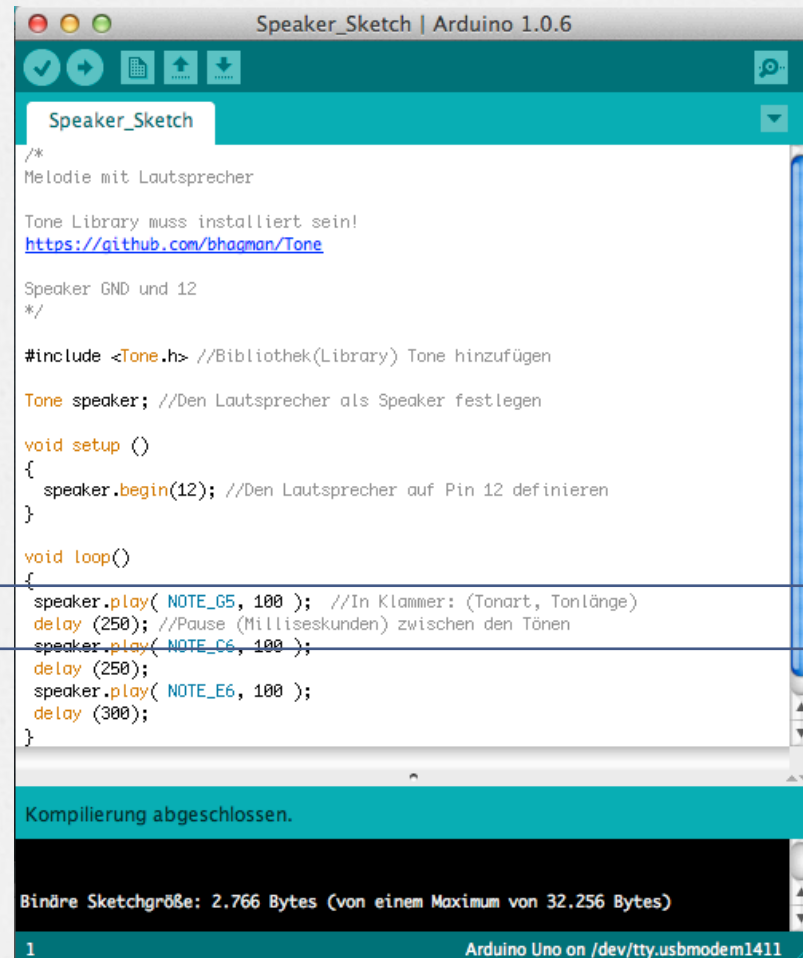
Sketch hochladen



fritzing

Physical Computing mit Arduino

Ordner Nr. 3: Töne erzeugen



The screenshot shows the Arduino IDE interface with a sketch named "Speaker_Sketch" open. The code is written in C++ and uses the Tone library. A blue box highlights the `speaker.play()` calls in the `loop()` function. The status bar at the bottom indicates "1" and "Arduino Uno on /dev/tty.usbmodem1411".

```
Speaker_Sketch | Arduino 1.0.6

Speaker_Sketch

/*
Melodie mit Lautsprecher

Tone Library muss installiert sein!
https://github.com/bogman/Tone

Speaker GND und 12
*/

#include <Tone.h> //Bibliothek(Library) Tone hinzufügen

Tone speaker; //Den Lautsprecher als Speaker festlegen

void setup ()
{
  speaker.begin(12); //Den Lautsprecher auf Pin 12 definieren
}

void loop()
{
  speaker.play( NOTE_G5, 100 ); //In Klammer: (Tonart, Tonlänge)
  delay (250); //Pause (Milliseskunden) zwischen den Tönen
  speaker.play( NOTE_C6, 100 );
  delay (250);
  speaker.play( NOTE_E6, 100 );
  delay (300);
}

Kompilierung abgeschlossen.

Binäre Sketchgröße: 2.766 Bytes (von einem Maximum von 32.256 Bytes)

1 Arduino Uno on /dev/tty.usbmodem1411
```


Physical Computing mit Arduino

```
#include <Tone.h> //Bibliothek(Library) Tone hinzufügen
Tone speaker; //Den Lautsprecher als Speaker festlegen

void setup ()
{
  speaker.begin(12); //Den Lautsprecher auf Pin 12 definieren
}

void loop()
{
  speaker.play( NOTE_G5, 100 ); //In Klammer: (Tonart, Tonlänge)
  delay (250); //Pause (Milliseskunden) zwischen den Tönen
  speaker.play( NOTE_C6, 100 );
  delay (250);
  speaker.play( NOTE_E6, 100 );
  delay (300);
}
```

Kompilieren abgeschlossen.

Globale Variablen verwenden 33 Bytes (1%) des dynamischen Speichers, 2.015 Bytes für lokale Variablen verbleiben. Das Maximum sind 2.048 Bytes.

2

Arduino Uno on /dev/cu.usbmodem1421

```
speaker.play( NOTE_G5, 100 ); //In Klammer: (Tonart, Tonlänge)
delay (250); //Pause (Milliseskunden) zwischen den Tönen
```

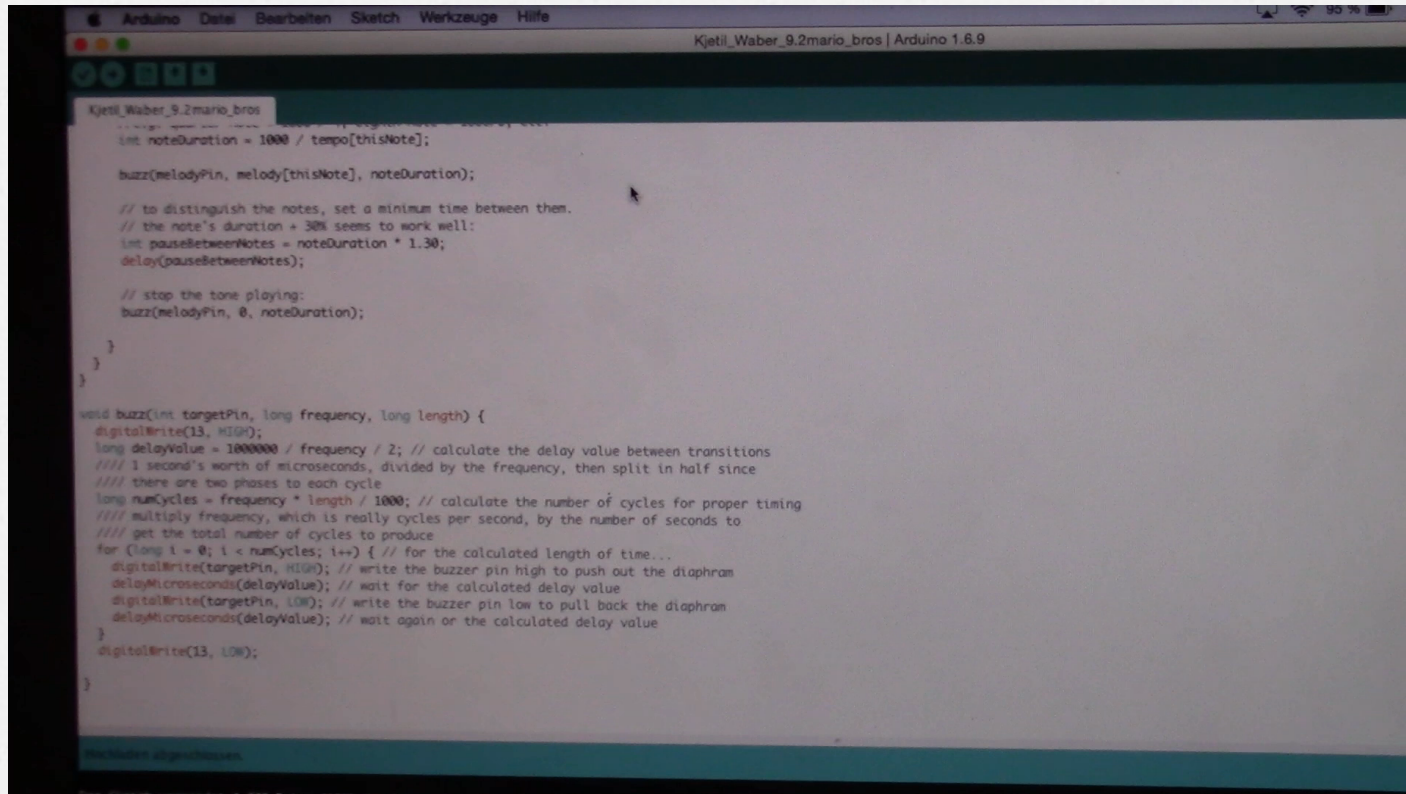
Abändern:

Tonart: Liste der Töne mit Frequenzen: <https://github.com/bhagman/Tone>

Tonlänge: Länge des Tons in Millisekunden

Pause: Dauer der Pause in Millisekunden

Physical Computing mit Arduino



```
Arduino Datei Bearbeiten Sketch Werkzeuge Hilfe
Kjetil_Waber_9.2mario_bros | Arduino 1.6.9

Kjetil_Waber_9.2mario_bros
// ...
int noteDuration = 1000 / tempo[thisNote];

buzz(melodyPin, melody[thisNote], noteDuration);

// to distinguish the notes, set a minimum time between them.
// the note's duration + 30% seems to work well:
int pauseBetweenNotes = noteDuration * 1.30;
delay(pauseBetweenNotes);

// stop the tone playing:
buzz(melodyPin, 0, noteDuration);
}
}

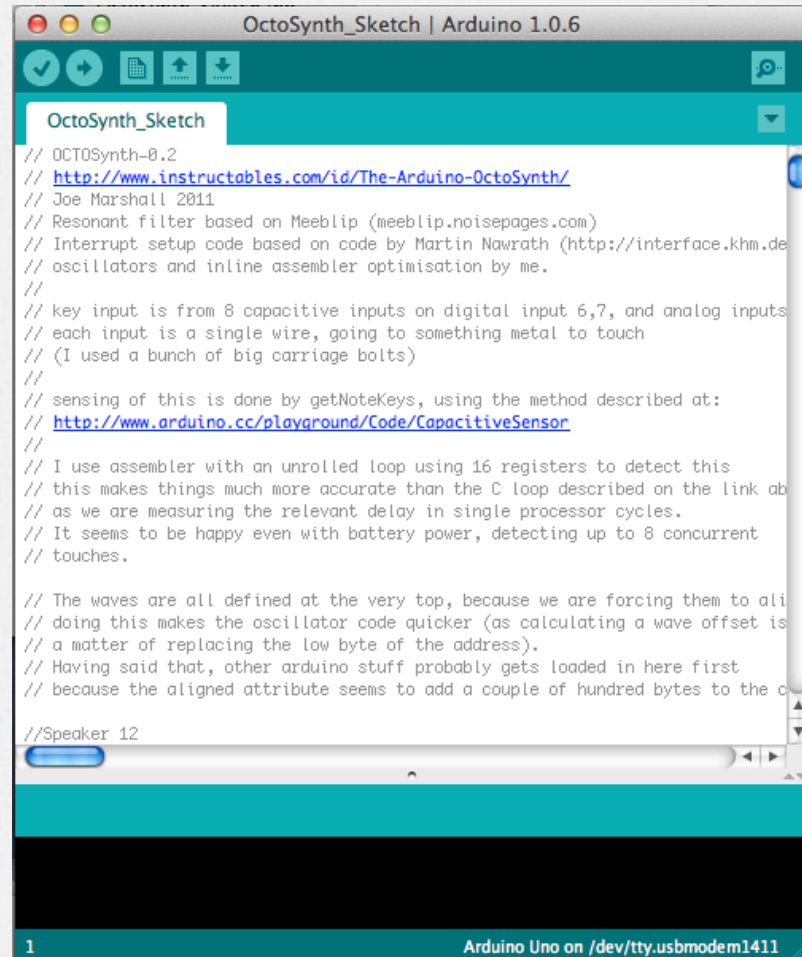
void buzz(int targetPin, long frequency, long length) {
  digitalWrite(13, HIGH);
  long delayValue = 1000000 / frequency / 2; // calculate the delay value between transitions
  // 1 second's worth of microseconds, divided by the frequency, then split in half since
  // there are two phases to each cycle
  long numCycles = frequency * length / 1000; // calculate the number of cycles for proper timing
  // multiply frequency, which is really cycles per second, by the number of seconds to
  // get the total number of cycles to produce
  for (long i = 0; i < numCycles; i++) { // for the calculated length of time...
    digitalWrite(targetPin, HIGH); // write the buzzer pin high to push out the diaphragm
    delayMicroseconds(delayValue); // wait for the calculated delay value
    digitalWrite(targetPin, LOW); // write the buzzer pin low to pull back the diaphragm
    delayMicroseconds(delayValue); // wait again on the calculated delay value
  }
  digitalWrite(13, LOW);
}
}
```

Rechtsklicken abgebrochen.

Physical Computing mit Arduino

Ordner Nr. 4: OctoSynth_Sketch

Sketch Verifizieren



```
OctoSynth_Sketch | Arduino 1.0.6

OctoSynth_Sketch
// OCTOSynth-0.2
// http://www.instructables.com/id/The-Arduino-OctoSynth/
// Joe Marshall 2011
// Resonant filter based on Meeblip (meeblip.noisepages.com)
// Interrupt setup code based on code by Martin Nawrath (http://interface.khm.de)
// oscillators and inline assembler optimisation by me.
//
// key input is from 8 capacitive inputs on digital input 6,7, and analog inputs
// each input is a single wire, going to something metal to touch
// (I used a bunch of big carriage bolts)
//
// sensing of this is done by getNoteKeys, using the method described at:
// http://www.arduino.cc/playground/Code/CapacitiveSensor
//
// I use assembler with an unrolled loop using 16 registers to detect this
// this makes things much more accurate than the C loop described on the link ab
// as we are measuring the relevant delay in single processor cycles.
// It seems to be happy even with battery power, detecting up to 8 concurrent
// touches.

// The waves are all defined at the very top, because we are forcing them to ali
// doing this makes the oscillator code quicker (as calculating a wave offset is
// a matter of replacing the low byte of the address).
// Having said that, other arduino stuff probably gets loaded in here first
// because the aligned attribute seems to add a couple of hundred bytes to the c

//Speaker 12

1 Arduino Uno on /dev/tty.usbmodem1411
```

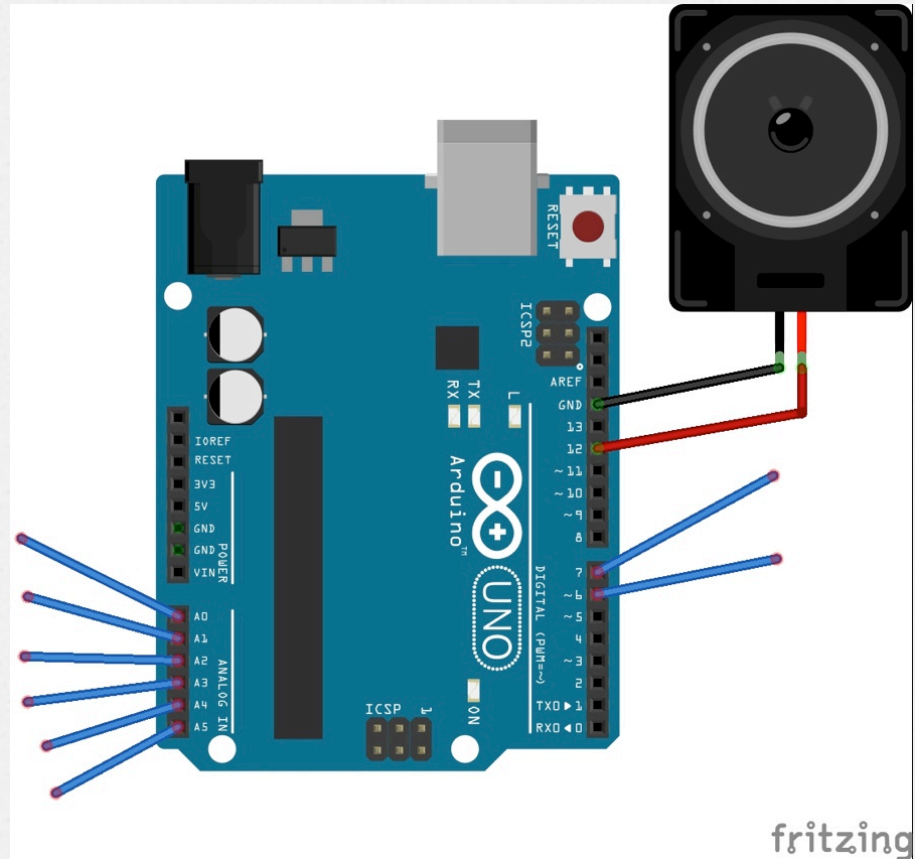
Physical Computing mit Arduino

8 Jumperkabel

6 7 Ao A1
A2 A3 A4 A5

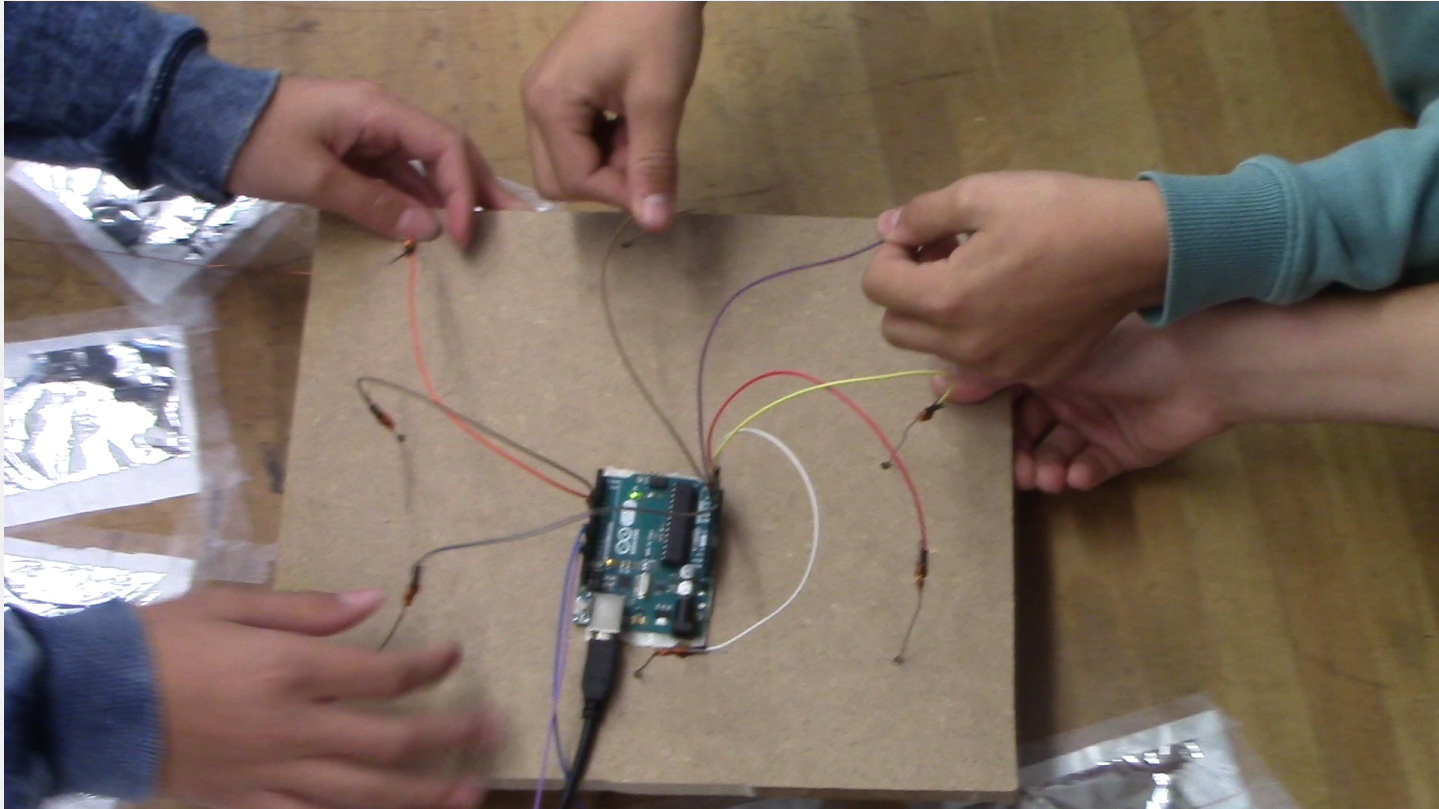
Lautsprecher
GND + 12

Arduino + PC
verbinden,
Sketch
hochladen



fritzing

Physical Computing mit Arduino



Physical Computing mit Arduino

Fragen?

Rückmeldung, Anregungen, Kritik

Werbung

Weiterbildung lernwerk bern

WS „Programmieren im Technischen Gestalten“

WS „Ein programmiertes Spiel bauen“